Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №5 по курсу**

**«Операционные системы»**

**ДИНАМИЧЕСКИЕ БИБЛИОТЕКИ**

Студент: Черненко Илья Денисович

Группа: М8О–206Б–18

Вариант: 27

Преподаватель: Соколов Андрей Алексеевич

Оценка: \_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2019.

**Содержание**

1. Постановка задачи
2. Общие сведения о программе
3. Общий метод и алгоритм решения
4. Основные файлы программы
5. Примеры работы
6. Вывод

**Постановка задачи**

## Цель работы

Целью является приобретение практических навыков в:

* Создание динамических библиотек
* Создание программ, которые используют функции динамических библиотек

## Задание

Требуется создать динамическую библиотеку, которая реализует определенный функционал. Далее использовать данную библиотеку 2-мя способами:

1. Во время компиляции (на этапе «линковки»/linking)
2. Во время исполнения программы, подгрузив библиотеку в память с помощью системных вызовов

В конечном итоге, программа должна состоять из следующих частей:

* Динамическая библиотека, реализующая заданных вариантом интерфейс;
* Тестовая программа, которая используют библиотеку, используя знания полученные на этапе компиляции;
* Тестовая программа, которая использует библиотеку, используя только местоположение динамической библиотеки и ее интерфейс.

Провести анализ между обоими типами использования библиотеки.

**Общие сведения о программе**

Программа компилируется из файла main.c. Также используется заголовочные файлы: stdio.h, stdbool.h, stdlib.h, string.h, ctype.h, vecmd5.h . В программе используются следующие системные вызовы:

1. **mmap** – отражает указанное кол-во байтов, начиная с некоторого смещения файла(определенного файловым описателем).
2. **munmap** – удаляет все отражения на указанную область памяти.
3. **sem\_post** – увеличивает (разблокирует) семафор, на который указывает аргумент функции. Если значение семафора после этого становится больше нуля, то другой процесс или нить заблокированная в вызове sem\_wait, проснётся и заблокирует семафор
4. **sem\_open –** создаёт новый семафор POSIX или открывает существующий семафор. Семафору присваивается имя, указанное в его аргументах.
5. **unlink** – удаляет имя и файл, на который он ссылается.
6. **sem\_wait** – уменьшает (блокирует) семафор, на который указывает *sem*. Если значение семафор больше нуля, то выполняется уменьшение и функция сразу завершается. Если значение семафора равно нулю, то вызов блокируется до тех пор, пока не станет возможным выполнить уменьшение (т. е., значение семафора не станет больше нуля), или пока не вызовется обработчик сигнала.

**Общий метод и алгоритм решения**.

Для реализации поставленной задачи необходимо:

1. Используя системный вызов mmap, используя файловый дескриптор fd(который ранее определяется функцией create\_tmp(), данные отражаются в память.
2. Используя системный вызов fork создать дочерний процесс.
3. В родительском процессе считывать данные cо стандартного потока и правильно распарсить подаваемую команду.
4. Как в родительском процессе данные считались, необходимо записать их в отображаемую память, с помощью sprintf.
5. Как только родительский процесс записал данные процесс считывает их, производит вычисления с деревом общего вида.
6. Дочерний процесс выводит результат используя write.

**Основные файлы программы**

**test1.c:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "lib/vecmd5.h"

int main() {

vector\* v = v\_create();

int c = 0;

while (scanf("%d", &c) && c) {

char value[33] = {'\0'};

if (c == 1) {

scanf("%s", value);

value[32] = '\0';

if (check(value)) {

v\_push(v, value);

} else {

printf("It's not an MD5\n");

}

} else if (c == 2) {

for (int i = 0; i < v\_get\_size(v); i++) {

printf("%s ", v\_get(v, i));

}

printf("\n");

} else if (c == 3) {

int pos = 0;

scanf("%d", &pos);

scanf("%s", value);

if (check(value)) {

v\_set(v, pos, value);

} else {

printf("It's not an MD5\n");

}

} else {

printf("Wrong command\n");

}

}

v\_delete(v);

return 0;

}

**test2.c**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <dlfcn.h>

#include "lib/vecmd5.h"

int main() {

vector\* (\*v\_create)();

void (\*v\_delete)(vector\* v);

void (\*v\_push)(vector\* v, md5 val);

void (\*v\_set)(vector\* v, int i, md5 val);

md5 (\*v\_get)(vector\* v, int i);

int (\*v\_get\_size)(vector\* v);

int (\*v\_get\_CAP)(vector\* v);

void (\*v\_set\_size)(vector\* v, int new\_size);

void (\*v\_set\_CAP)(vector\* v, int new\_size);

bool (\*check)(char\* val);

char \*err;

void \*libHandle;

libHandle = dlopen("../cmake-build-debug/libos\_lab\_5\_lib.so", RTLD\_LAZY);

if (!libHandle) {

fprintf(stderr, "%s\n", dlerror());

exit(-1);

}

v\_create = dlsym(libHandle, "v\_create");

v\_delete = dlsym(libHandle, "v\_delete");

v\_push = dlsym(libHandle, "v\_push");

v\_set = dlsym(libHandle, "v\_set");

v\_get = dlsym(libHandle, "v\_get");

v\_get\_size = dlsym(libHandle, "v\_get\_size");

v\_get\_CAP = dlsym(libHandle, "v\_get\_CAP");

v\_set\_size = dlsym(libHandle, "v\_set\_size");

v\_set\_CAP = dlsym(libHandle, "v\_set\_CAP");

check = dlsym(libHandle, "check");

err = dlerror();

if(err) {

fprintf(stderr, "%s\n", err);

exit(-1);

}

vector\* v = v\_create();

int c = 0;

while (scanf("%d", &c) && c) {

char value[33] = {'\0'};

if (c == 1) {

scanf("%s", value);

value[32] = '\0';

if ((\*check)(value)) {

(\*v\_push)(v, value);

} else {

printf("It's not an MD5\n");

}

} else if (c == 2) {

for (int i = 0; i < (\*v\_get\_size)(v); i++) {

printf("%s ", (\*v\_get)(v, i));

}

printf("\n");

} else if (c == 3) {

int pos = 0;

scanf("%d", &pos);

scanf("%s", value);

if ((\*check)(value)) {

(\*v\_set)(v, pos, value);

} else {

printf("It's not an MD5\n");

}

} else {

printf("Wrong command\n");

}

}

(\*v\_delete)(v);

dlclose(libHandle);

return 0;

}

**vecmd5.h**

#ifndef OS\_LAB\_5\_VECMD5\_H

#define OS\_LAB\_5\_VECMD5\_H

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <stdbool.h>

#include <ctype.h>

typedef char\* md5;

typedef struct {

md5\* body;

int size;

int CAP;

} vector;

vector\* v\_create();

void v\_delete(vector\* v);

void v\_push(vector\* v, md5 val);

void v\_set(vector\* v, int i, md5 val);

md5 v\_get(vector\* v, int i);

int v\_get\_size(vector\* v);

int v\_get\_CAP(vector\* v);

void v\_set\_size(vector\* v, int new\_size);

void v\_set\_CAP(vector\* v, int new\_size);

bool check(char\* val);

#endif //OS\_LAB\_5\_VECMD5\_H

**vecmd5.c**

#include "vecmd5.h"

vector\* v\_create() {

vector\* v = (vector\*)malloc(sizeof(vector));

v->body = NULL;

v->size = 0;

v->CAP = 1;

return v;

}

void v\_delete(vector\* v) {

for (int i = 0; i < v\_get\_size(v); i++) {

free(v->body[i]);

}

free(v->body);

free(v);

}

void v\_push(vector\* v, md5 val) {

int size = v\_get\_size(v);

if (size + 1 == v\_get\_CAP(v)) {

v->CAP \*= 2;

v\_set\_CAP(v, v->CAP);

v->body[size] = (md5) malloc(33);

strcpy(v->body[size], val);

v\_set\_size(v, size + 1);

} else {

v->body[size] = (md5) malloc(33);

strcpy(v->body[size], val);

v\_set\_size(v, size + 1);

}

}

void v\_set(vector\* v, int i, md5 val) {

if (i > v\_get\_size(v)) {

printf("Out of bounds\n");

} else if (v\_get\_size(v) + 1 == v\_get\_CAP(v)) {

v->CAP \*= 2;

v\_set\_CAP(v, v->CAP);

int size = v\_get\_size(v);

v\_set\_size(v, size + 1);

v->body[i] = (md5)malloc(33);

strcpy(v->body[i], val);

} else if (i == v\_get\_size(v)) {

v->body[i] = (md5) malloc(33);

strcpy(v->body[i], val);

v\_set\_size(v, i + 1);

} else {

strcpy(v->body[i], val);

}

}

md5 v\_get(vector\* v, int i) {

return v->body[i];

}

int v\_get\_size(vector\* v) {

return v->size;

}

int v\_get\_CAP(vector\* v) {

return v->CAP;

}

void v\_set\_size(vector\* v, int new\_size) {

v->size = new\_size;

}

void v\_set\_CAP(vector\* v, int new\_size) {

md5\* reBody = (md5\*)realloc(v->body, new\_size \* sizeof(md5));

v->body = reBody;

}

bool check(char\* val) {

for (int i = 0; i < 32; i++) {

if (!isalpha(val[i]) && !isdigit(val[i])) {

return false;

}

}

return true;

}

**Примеры работы**

**test1:**

/home/zebr/os\_lab\_4-master/src/cmake-build-debug/os\_lab\_4

add s 3434

prt

3434

add s 999

PRT

invalid command

prt

3434

999

ext

Process finished with exit code 0

**test2:**

/home/zebr/os\_lab\_4-master/src/cmake-build-debug/os\_lab\_4

add s 34

add s 22

add sb 9

prt

34

22

9

add ss 0

prt

34

22

0

9

rmv ssb

its root

prt

empty tree

ext

Process finished with exit code 0

**test3:**

/home/zebr/os\_lab\_4-master/src/cmake-build-debug/os\_lab\_4

add s 324

prt

324

add s 929

prt

324

929

add s 223

prt

324

223

929

add ssb 31

prt

324

223

929

add sb 0

prt

324

223

0

929

add sbs 34

prt

324

223

0

34

929

ext

Process finished with exit code 0

**Вывод**

Обрел навыки работы с динамическими библиотеками и их созданием, ознакомился с устройством работы семафоров. Ознакомился с принципами работы (), и применил этот системный вызов для передачи данных дочернему процессу. Изучил принципы работы при создании временных файлов в ОС Linux.